FermiGrid Design Note

For

Software Release Management
and
Software Acceptance Process

Keith Chadwick
16-May-2008

Abstract:

This document describes the software acceptance process that is used by FermiGrid

Document Revision History:

Version	Date	Author	Comments
0.1	12-Feb-08	Keith Chadwick	Initial version.
0.2	05-May-08	Keith Chadwick	Clean up formatting and content.
0.3	16-May-08	Eileen Berman	Small changes to wording.
0.4	16-May-08	Keith Chadwick	More wording tweaks.
0.5	16-May-08	Igor Mandrichenko	Modified some wording, added comments

Introduction:

Effective operation and management of FermiGrid requires a strict software release management and software acceptance process in order to insure operation of the FermiGrid services.

Additional goals of this process include:

- Effective tests at production scale.
- Ease administration effort by operations group.
- Reduce need to contact the developer(s).

In order to support these goals, software project managers will need to:

- Incorporate time to transition to operations in WBS.
- Incorporate documentation (install, operate, capture debug information, upgrade).

We also need to distinguish between three categories of software releases:

- Emergency bug fixes to restore existing functionality.
- Normal routine bug fixes of existing functionality.
- Extensions to existing functionality.

Developers need to think about running their code on a system that is providing other services and cannot "just" be rebooted.

Prioritize needed enhancements.

Perform a survey to understand where we are today with respect to this process.

Multiple forms of software packaging/distribution methods shall be accepted. Examples include pacman, RPM, ups/upd and bare tarballs.

Software Acceptance Environment:

In general, there shall be no root access to the production software or service instance by developers (other than explicitly enabled by FermiGrid personnel during problem diagnosis and/or resolution).

Software Documentation:

FermiGrid personnel shall receive a documentation package from the service developers that includes:

1. How to install and/or upgrade the service.
2. How to revert back to a prior version of the service if necessary.
3. How to start the service.
4. How to shutdown the service.
5. How to test core (or essential) service functionality (ideally a full regression test).
6. How to monitor core (or essential) service functionality.
7. A description of the service log locations and logging options.
8. A list of information to capture in the event of problems (debug script?) etc.
9. Other information that is necessary to insure service operation.

It is expected that all of the above documents are "living documents" – they will be developed, revised and enhanced as part of the service development and deployment lifecycle.

Software Development/Integration/Production Procedure:

The service development team must provide a well-defined process (including version release procedure and regression test suite) as detailed in the process below.

All source code running on production systems must be 'tagged' such that it can be identified in the source code management system used by the developers.

Ivm: Although previous paragraph describes commonly accepted as necessary and useful software development practice, I do not see why this paragraph belongs in this document. E.g. if no code management system was used by developers, or code management system they use does not support tagging, why the service is not acceptable for FermiGrid to run ?

Note 1 - The regression test suite on the [production, integration, development] system corresponds to the [production, integration, development] software version.

Note 2 - The process below is the process used by FermiGrid to test, integrate and deploy the SAZ package.

1. Development of the service takes place on development system(s) that are either operated by the developers or otherwise operated where the developers have root access.

2. Development version service passes the “regression test suite” that is defined and provided by the developers. This test suite may be a manual process or (ideally) an automated process. The regression test suite must be capable of testing core or critical portions of service functionality as well as major new features introduced in new release and must also provide a “load” or “stress” test functions, when applicable and practical. If the regression test suite is not automated then it must be explicitly documented to assure that all tests have been performed each and every time that a release is cut. If it is not documented, then it never happened. Developers are responsible for defining test acceptance criteria.

3. The developers perform a formal software version cut and provide FermiGrid with pachakge in appropriate format (see Introduction).

4. FermiGrid personnel install this version on the test/integration system(s) using the instructions provided by the developers. Note: Developers may assist in the installation, but any such "assistance" must be immediately entered into the documentation package by the developers.

5. Test/integration installation passes regression test suite. FermiGrid personnel run the regression tests with the assistance of the developers where necessary. Any “unusual” assistance by the developers must be immediately entered into the service documentation package.

6. Formal acceptance of version release to production by FermiGrid management. (this has to be specified: based on what criteria?)

7. Service version installation on production system(s) by FermiGrid personnel.

8. Production installation passes regression test suite.

9. Formal release to operation.

Software Support Infrastructure:

Developers must supply a support infrastructure for “expert” level questions. In the case of packages that are distributed by the VDT, then this is the case.